8051 Micro-controller Lab Manual

Table of Contents

Addition of two 8-bit numbers

Address	Mnemonics	Opcode	Operands	Comment
		MOV A,	#O5H	05H is moved to A
		MOV R1,	#15H	15H is moved to R1
		ADD A,R1		Add the value of A and R1
		MOV DPTR,	#9000H	DPTR is initialized data pointer is set to 9000H
		MOVX @DPTR,A		Value of A is moved to DPTR
		JMP	OOOO	Stop

Addition of two 16-bit numbers

Address	**Mnemonics**	**Opcode**	**Operands**	**Comments**
		MOV A,	#34H	34 is moved to A
		ADD A,	#62H	62 is moved to A
		MOV DPTR	#9000H	DPTR is initialized data pointer is set to 9000H
		MOVX @DPTR,A		Value of A is moved to DPTR
		MOV A,	#12	12 is moved to A
		ADDC A,	#24	Add 24 to the value of A
		INC DPTR		Increment the location of DPTR
		MOVX @DPTR,A		Value of A is moved to DPTR
		JMP	0000	Stop

SUBTRACTION OF 8-BIT NUMBER

Address	Mnemonics	Opcode	Operands	Comments
		MOV R1,	#05H	05 is moved to R1
		MOV A	#15H	15 is moved to A
		CLR C		Clear the carry flag
		SUBB A,R1		Subtract 15 from 05
		MOV DPTR,	#9000H	Initialize the data pointer is set to 9000H
		MOVX@DPTR,A		Value of A is moved to 9000H
		JMP	0000	Stop

SUBTRACTION OF 16-BIT NUMBER

Address	Mnemonics	Opcode	Operands	Comments
		MOV A	#09H	09H is moved to A
		SUBB A	#06H	06H is subtracted from A
		MOV DPTR	#9000H	Initialize the DPTR, data pointer is set to900H
		MOV@DPTR,A		Move the value of A to DPTR
		MOV A	#07H	07H is moved to A
		SUBB A	#04H	04H is subtracted from A
		INC DPTR		DPTR is incremented
		MOVX @ DPTR,A		Move the value of A to DPTR
		JMP	0000	Stop

Multiplication of two 8-bit numbers

Address	Mnemonics	Opcode	Operands	Comments
		MOV A,	#02H	Move 02 to accumulator
		MOV B,	#04H	Move 04 to Reg B
		MUL AB		Multiply the value of A and B
		MOV DPTR,	#9000H	Initialize the DPTR and DPTR is set to 9000H
		MOV X@ DPTR,A		Value of A is moved to 9000H
		JMP	00 00 00	Stop

Division of 8-bit number

Address	Mnemonics	Opcode	Operands	Comments
		MOV A,	#65H	65 is moved to A
		MOV B,	#05H	05 is moved to A
		DIV AB		Divide A by B
		MOV DPTR,	9000H	DPTR is initialized data pointer is set to 9000H
		MOVX @DPTR,A		Value of A is moved to DPTR
		INC DPTR		Value of DPTR is incremented
		MOV A,B		Value of A is moved to B
		MPVX @DPTR,A		DPTR is initialized data pointer is set to 9000H
		JMP	0000	Stop

2's complement of 8-bit number

Address	Mnemonics	Opcode	Operands	Comments
		MOV A,	#05H	05H is moved to A
		CPL A,		A is complimented
		ADDA	#01H	01 is added to A
		MOV DPTR,	#9000H	Initialize the DPTR, Data pointer is set to 9000H
		MOVX @DPTR,A		Move the value of A to DPTR
		JMP	0000	Stop

2's complement of 16-bit number

Address	Mnemonics	Opcode	Operands	Comments
		MOV A,	#2A	Move 2A to accumulator
		CPL A		Compliment A
		MOV R1,A		Move the value of A to R1
		MOV A	#3B	Mov 3B to A
		CPL A		Compliment A
		MOV R2,A		Move the value of A to R2
		MOV A,R1		Move value of R1 to A
		ADD A,	#01	ADD 01 to A
		MOV DPTR,	#9000H	Initialize the DPTr data pointer is set to 9000H
		MOVX @ DPTR, A		Move the value of A to DPTR
		MOV A,R2		
		ADDC A,	#00H	ADD 00H to A
		INC DPTR		Increment DPTR
		MOVX @DPTR,A		Move the value of A to DPTR
		LJMP	0000	Stop

Largest of N numbers

```
ORG 0000H          ; Origin, start code at address 0000H
MOV R0, #00H       ; Initialize R0 with 00H (R0 will point to data in memory)
MOV A, @R0         ; Move the value at the memory location pointed by R0 to
the accumulator (A)
MOV R2, A          ; Copy the value in A (accumulator) to R2 (R2 used as a
counter)
DEC R2             ; Decrement the value in R2 (R2 = R2 - 1)

INC R0             ; Increment R0 to point to the next memory location

BACK: MOV A, @R0   ; Move the value at memory pointed by R0 into A
CJNE A, B, LOOP    ; Compare A with B, if not equal, jump to LOOP
JMP NEXT           ; If A equals B, jump to NEXT

LOOP: JC LOOP1     ; If the carry flag is set, jump to LOOP1
MOV B, A           ; Move the value in A to B

LOOP1: INC R0      ; Increment R0 to point to the next memory location
DJNZ R2, BACK      ; Decrement R2, if not zero, jump back to BACK

NEXT: MOV 60H, B   ; Move the value of B into memory address 60H
END          ; End of program
```

COUNT NUMBER OF ONE’S AND ZERO’S IN A NUMBER

```
ORG 0000H           ; Set the origin at address 0000H

MOV R2, #00H        ; Initialize R2 to store the count of '1's
MOV R3, #00H        ; Initialize R3 to store the count of '0's
MOV R1, #08H        ; Set R1 to 8, as we are working with an 8-bit
        number
MOV R0, #06H        ; Load the number 06H into R0 (this is the
        number whose bits will be counted)
MOV A, R0           ; Move the value in R0 (06H) to the
        accumulator (A)

BACK: RRC A         ; Rotate the accumulator right through
        the carry flag
JC SKIP       ; If carry is set (bit was 1), jump to SKIP
INC R3        ; Increment R3 to count the '0' bit
AJMP LAST           ; Jump to LAST

SKIP: INC R2        ; Increment R2 to count the '1' bit

LAST: DJNZ R1, BACK  ; Decrement R1 (bit counter), if not zero, go
        back to BACK

END           ; End of program
```

SQUARE NUMBER USING LOOKUP TABLE

```
 ORG 0000H          ; Set program origin at address 0000H

MOV DPTR, #300H          ; Load the starting address of the square lookup
table (300H) into DPTR
MOV A, 60H          ; Move the value from memory location 60H into A
(assumed to contain the number to square)
MOVC A, @A+DPTR          ; Use the value in A as an index to retrieve the
square from the lookup table (A = A + DPTR)
MOV 70H, A          ; Store the result (square of the number) in memory
location 70H

ORG 300H            ; Set origin at 300H for the lookup table
SQR_TABLE:          ; Define the square lookup table
DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81  ; Values are squares of numbers 0-9

END                 ; End of program
```

SWITCH STATUS

```
ORG 0000H           ; Set program origin at address 0000H

        SETB P0.0             ; Set bit P0.0
(initialize switch on Port 0, pin 0 to high state)
        CLR P2.0              ; Clear bit P2.0 (turn
off an LED or other device connected to Port
2, pin 0)

        AGAIN: JNB P0.0, NEXT ; Check the
status of P0.0 (switch). If P0.0 is low (not
pressed), jump to NEXT
                SETB P2.0     ; If P0.0 is high
(switch pressed), set P2.0 (turn on the
LED/device)
                SJMP AGAIN          ; Jump back to
AGAIN to keep monitoring the switch status

        NEXT: CLR P2.0        ; If P0.0 is low
(switch not pressed), clear P2.0 (turn off the
LED/device)
                SJMP AGAIN          ; Keep
checking the switch status in an infinite loop

        END             ; End of the program
```

TIME DELAY USING TIMER

```
#include <reg51.h>   // Include header file for 8051 microcontroller

void DELAY(void);      // Function prototype for delay

void main(void) {
   while (1) {    // Infinite loop
            P1 = 0x55;   // Send 0x55 (01010101) to Port 1 (LED pattern)
            DELAY();       // Call delay function
            P1 = 0xAA;   // Send 0xAA (10101010) to Port 1 (LED pattern)
            DELAY();       // Call delay function
   }
}

void DELAY(void) {
   TMOD = 0x01;        // Set timer mode: Timer 0, Mode 1 (16-bit timer mode)
   TH0 = 0x4B;           // Load higher byte of timer with 4B (for delay)
   TL0 = 0xFE;   // Load lower byte of timer with FE (for delay)
   TR0 = 1;        // Start Timer 0

   while (TF0 == 0); // Wait for Timer 0 overflow (TF0 flag set)

   TR0 = 0;        // Stop Timer 0
   TF0 = 0;        // Clear Timer 0 overflow flag
}
```

Arrange the number in ascending order

```
ORG 0000H              ; Program origin at address 0000H
MOV R0, #09H           ; Initialize R0 with 9 (outer loop counter)

AGAIN: MOV DPTR, #2000H  ; Initialize DPTR to point to external memory
starting at 2000H
    MOV R1, #09H       ; Initialize R1 with 9 (inner loop counter)

BACK:  MOV A, DPL      ; Load lower byte of DPTR (DPL) into A
    MOVX A, @DPTR      ; Move the external memory byte at DPTR into A
(using MOVX for external memory)
    MOV B, A           ; Copy the value in A into B (temporary storage)

    INC DPTR           ; Increment DPTR to point to the next memory
location
    MOVX A, @DPTR      ; Move the next byte from external memory at DPTR
into A
    CJNE A, B, NEXT   ; Compare A (current value) with B (previous value),
jump to NEXT if not equal

    AJMP SKIP          ; If A equals B, skip to SKIP (no operation needed)

NEXT:  JC SKIP         ; If carry is set (indicating a negative result), jump to
SKIP
    MOV DPL, R2        ; Load the lower byte of DPTR with the value in R2
(presumably modifying DPTR)
    MOVX @DPTR, A      ; Move the current value in A into external memory
at DPTR

    INC DPTR           ; Increment DPTR to point to the next memory
location
    MOV A, B           ; Load A with the value previously stored in B
    MOVX @DPTR, A      ; Store the value in A (which was originally in B) into
the new DPTR location

SKIP:  DJNZ R1, BACK   ; Decrement R1, if not zero, jump back to BACK
(inner loop)
    DJNZ R0, AGAIN     ; Decrement R0, if not zero, jump back to AGAIN
(outer loop)

END              ; End of the program
```